Claim Amendments

Please amend the claims as indicated:

1. (original) A method for transferring a data set between a transfer-initiating subsystem and a target via an intermediate subsystem, a transfer-executing subsystem, and a channel, where:

the transfer-initiating subsystem follows a transfer protocol in which the data set is transferred as a series of packets, and transfer of at least one subsequent packet is contingent on completed transfer of at least one previous packet;

data transfer between the transfer-initiating subsystem and the transfer-executing subsystem takes place via the intermediate subsystem;

the transfer-executing subsystem initiates data transfer via the channel upon receipt of a transfer request and of identification of the data set to be transferred;

the method comprising:

within the intermediate subsystem,

A) receiving a first one of the packets from the transfer-initiating subsystem;

B) signaling to the transfer-initiating subsystem that the transfer of the first packet is still pending, so that the transfer-initiating subsystem at least temporarily delays submission of subsequent packets to the intermediate subsystem;

C) issuing to the transfer-executing subsystem a transfer request and identifying for the transfer-executing subsystem the data set to be transferred, whereupon the transfer-executing subsystem attempts transfer of the data set;

D) upon sensing a completion signal from the transfer-executing subsystem indicating completed transfer of the data set,

i) signaling to the transfer-initiating subsystem completed transfer of the first packet;

ii) receiving subsequent packets from the transfer-initiating subsystem and, for each, signaling to the transfer-initiating subsystem completed transfer of each subsequent packet.

2. A method as in claim 1, in which:

A) the direction of transfer of the data set is from the transfer-initiating subsystem to the target; and

B) the transfer-initiating subsystem stores the data set in a memory space accessible to the intermediate subsystem;

C) the method further comprising the following steps:

i) creating, within the intermediate subsystem, and independently of action by the transfer-initiating subsystem, a copy of the data set by accessing the memory space in which the transfer-initiating subsystem has stored the data set; and

ii) identifying for the transfer-executing subsystem the copy of the data set as the data set to be transferred.

3. (original) A method as in claim 2, in which the transfer-initiating subsystem derives, for the data set stored in the memory space, a plurality of descriptors, each descriptor specifying at least a memory location of a corresponding sub-set of the data set, and in which packets are generated at least partially based on the descriptors, the method further comprising the following step:

in the intermediate subsystem, creating the copy of the data set by evaluating the descriptors. (original)

4. (original) A method as in claim 2, further including the following steps:

in the intermediate subsystem, upon sensing the completion signal from the transfer-executing subsystem, determining whether the transfer-initiating subsystem has changed the data set since the time when the first packet was received and, if so, creating and submitting to the transfer-executing subsystem for transfer to the target a copy of the changed data set.

5. (original) A method as in claim 4, in which the step of determining whether the transfer-initiating subsystem has changed the data set comprises comparing the copy of the data set with a possibly changed data set currently designated for transfer by the transfer-initiating subsystem.

6. (original) A method as in claim 4, in which the transfer-initiating subsystem derives, for the data set stored in the memory space, a plurality of descriptors, each descriptor specifying at least a memory location of a corresponding sub-set of the data set, and in which packets are generated at least partially from the descriptors, the method further comprising the following steps:

storing for access by the intermediate subsystem a copy of the descriptors corresponding to the copy of the data set currently being transferred by the transfer-executing subsystem; and

in the intermediate subsystem, determining whether the transfer-initiating subsystem has changed the data set by comparing the descriptors corresponding to the data set currently being transferred by the transfer-executing subsystem with the descriptors of a possibly changed data set currently designated for transfer by the transfer-initiating subsystem.

7. (original) A method as in claim 4, in which:

the transfer-initiating subsystem derives and stores, at respective descriptor memory addresses, a plurality of descriptors, each descriptor identifying a corresponding sub-set of the data set, the method further comprising the following steps:

for each descriptor, creating an outrecord identifying the corresponding descriptor memory address; and

determining whether the transfer-initiating subsystem has changed the data set by comparing the descriptor memory addresses identified in each outrecord with the addresses in memory of the descriptors most recently created by the transfer-initiating subsystem for the data set.

8. (original) A method as in claim 1, in which the direction of transfer of the data set is from the target to the transfer-initiating subsystem, the method further comprising the following steps:

A) creating a buffer within the intermediate subsystem;

B) storing in the buffer the data set transferred from the target; and

C) upon completed transfer and storage of the data set, and upon receipt by the intermediate subsystem of each packet from the transfer-initiating subsystem, copying a corresponding portion of the data set to the packet

9. (original) A method as in claim 8, in which the transfer-initiating subsystem derives, for the data set to be transferred from the target, a plurality of descriptors, each descriptor specifying at least a size of a corresponding subset of the data set to be transferred, the method further comprising creating the buffer as a function of the plurality of descriptors, the size of each portion of the data set copied to its respective packet corresponding to the size of the subset specified by the corresponding descriptor.

10. (original) A method as in claim 1, in which the channel between the transfer-executing subsystem and the target is a first channel, further comprising the steps of emulating a second channel in the intermediate subsystem and directing all packet transfer between the transfer-initiating subsystem and the target to the emulated, second channel.

11. (original) A method as in claim 1, in which the transfer-initiating subsystem is a software-implemented computer.

12. (original) A method as in claim 11, in which the transfer-initiating subsystem is a virtual machine.

13. (original) A method as in claim 1, in which:

the channel is a Universal Serial Bus (USB);

the target is a USB device and respective pipe; and

both the transfer-initiating subsystem and the transfer-executing subsystem generate and transfer packets according to the USB protocol.

14. (original) A method as in claim 1, further comprising the step of transferring data between the transfer-executing subsystem and the target according to a different protocol than the transfer protocol that the transfer-initiating subsystem follows.

15. (original) A method for transferring a data set between a virtual machine and a target via a virtual machine monitor, a host interface, and a channel, where:

the virtual machine follows a transfer protocol in which the data set is transferred as a series of packets, and transfer of at least one subsequent packet is contingent on completed transfer of at least one previous packet;

data transfer between the virtual machine and the host interface takes place via the virtual machine monitor;

the host interface initiates data transfer via the channel upon receipt of a transfer request and of identification of the data set to be transferred, and of identification of the target;

the method comprising:

within the virtual machine monitor,

A) receiving a first one of the packets from the virtual machine;

B) signaling to the virtual machine that the transfer of the first packet is still pending, so that the virtual machine at least temporarily delays submission of subsequent packets to the virtual machine monitor;

C) issuing to the host interface a transfer request and identifying for the host interface the data set to be transferred, whereupon the host interface attempts transfer of the data set;

D) upon sensing a completion signal from the host interface indicating completed transfer of the data set,

i) signaling to the virtual machine completed transfer of the first packet;

ii) receiving subsequent packets from the virtual machine and, for each, signaling to the virtual machine completed transfer of each subsequent packet.

16. (original) A method as in claim 15, in which:

A) the direction of transfer of the data set is from the virtual machine to the target; and

B) the virtual machine stores the data set in a memory space accessible to the virtual machine monitor;

C) the method further comprising the following steps:

i) creating, within the virtual machine monitor, and independently of action by the virtual machine, a copy of the data set by accessing the memory space in which the virtual machine has stored the data set; and

ii) identifying for the host interface the copy of the data set as the data set to be transferred.

17. (original) A method as in claim 16, in which the virtual machine derives, for the data set stored in the memory space, a plurality of descriptors, each descriptor specifying at least a memory location of a corresponding sub-set of the data set, and in which packets are generated at least partially based on the descriptors, the method further comprising the following step:

in the virtual machine monitor, creating the copy of the data set by evaluating the descriptors. (original)

18. (original) A method as in claim 16, further including the following steps:

in the virtual machine monitor, upon sensing the completion signal from the host interface, determining whether the virtual machine has changed the data set since the time when the first packet was received and, if so, creating and submitting to the host interface for transfer to the target a copy of the changed data set.

19. (original) A method as in claim 18, in which the step of determining whether the virtual machine has changed the data set comprises comparing the copy of the data set with a possibly changed data set currently designated for transfer by the virtual machine.

20. (original) A method as in claim 18, in which the virtual machine derives, for the data set stored in the memory space, a plurality of descriptors, each descriptor specifying at least a memory location of a corresponding sub-set of the data set, and in which packets are generated at least partially from the descriptors, the method further comprising the following steps:

storing for access by the virtual machine monitor a copy of the descriptors corresponding to the copy of the data set currently being transferred by the host interface; and

in the virtual machine monitor, determining whether the virtual machine has changed the data set by comparing the descriptors corresponding to the data set currently being transferred by the host interface copy with the descriptors of a possibly changed data set currently designated for transfer by the virtual machine.

21. (original) A method as in claim 15, in which the direction of transfer of the data set is from the target to virtual machine, the method further comprising the following steps:

A) creating a buffer within the virtual machine monitor;

B) storing in the buffer the data set transferred from the target; and

C) upon completed transfer and storage of the data set, and upon receipt by the virtual machine monitor of each packet from the virtual machine, copying a corresponding portion of the data set to the packet.

22. (original) A method as in claim 21, in which the virtual machine derives, for the data set to be transferred from the target, a plurality of descriptors, each descriptor specifying at least a size of a corresponding subset of the data set to be transferred, the method further comprising creating the buffer as a function of the plurality of descriptors, the size of each portion of the data set copied to its respective packet corresponding to the size of the subset specified by the corresponding descriptor.

23. (original) A method as in claim 15, in which the channel between the host interface and the target is a first channel, further comprising the steps of emulating a second channel in the virtual machine monitor and directing all packet transfer between the virtual machine and the target to the emulated, second channel.

24. (original) A method as in claim 15, in which:

the channel is a Universal Serial Bus (USB);

the target is a USB device and respective pipe; and

both the virtual machine and the host interface generate and transfer packets according to the USB protocol.

25. canceled

26. canceled

27. (original) A method for transferring a data set from a transfer-initiating subsystem to a target via an intermediate subsystem, a transfer-executing subsystem, and a channel, where:

A) the transfer-initiating subsystem

i) stores the data set in a memory space accessible to the intermediate subsystem;

ii) derives, for the data set stored in the memory space, a plurality of descriptors, each descriptor specifying at least a memory location of a corresponding sub-set of the data set;

iii) follows a transfer protocol in which the data set is transferred as a series of packets;

iv) generates the packets at least partially based on the descriptors; and

v) transfers at least one subsequent packet contingent on completed transfer of at least one previous packet;

B) data transfer from the transfer-initiating subsystem to the transfer-executing subsystem takes place via the intermediate subsystem;

C) the transfer-executing subsystem initiates data transfer via the channel upon

receipt of a transfer request and of identification of the data set to be transferred;

the method comprising:

D) within the intermediate subsystem,

i) creating, independently of action by the transfer-initiating subsystem, a copy of the data set by evaluating the descriptors and by accessing the memory space in which the transfer-initiating subsystem has stored the data set; and

ii) identifying for the transfer-executing subsystem the copy of the data set as the data set to be transferred and issuing the transfer request to the transfer-executing subsystem;

iii) receiving a first one of the packets from the transfer-initiating subsystem;

iv) signaling to the transfer-initiating subsystem that the transfer of the first packet is still pending, so that the transfer-initiating subsystem at least temporarily delays submission of subsequent packets to the intermediate subsystem;

v) issuing to the transfer-executing subsystem a transfer request and identifying for the transfer-executing subsystem the data set to be transferred, whereupon the transfer-executing subsystem attempts transfer of the data set to the target; and

vi) upon sensing a completion signal from the transfer-executing subsystem indicating completed transfer of the data set,

a) signaling to the transfer-initiating subsystem completed transfer of the first packet; and

b) receiving subsequent packets from the transfer-initiating subsystem and, for each, signaling to the transfer-initiating subsystem completed transfer of each subsequent packet.

28. (original) A system for transferring a data set between a transfer-initiating subsystem and a target via a transfer-executing subsystem and over a channel, where:

A) the transfer-initiating subsystem follows a transfer protocol in which the data set is transferred as a series of packets, and transfer of at least one subsequent packet is contingent on completed transfer of at least one previous packet;

B) the transfer-executing subsystem initiates data transfer via the channel upon receipt of a transfer request and of identification of the data set to be transferred;

the system comprising:

C) an intermediate subsystem

i) forming an interface between the transfer-initiating subsystem and the transfer-executing subsystem,

ii) forming means for identifying for the transfer-executing subsystem the data set to be transferred, whereupon the transfer-executing subsystem attempts transfer of the data set;

iii) and including:

a) an emulated channel forming means for receiving the packets from the transfer-initiating subsystem and for signaling to the transfer-initiating subsystem that the transfer of a first packet is still pending, so that the transfer-initiating subsystem at least temporarily delays submission of subsequent packets to the intermediate subsystem;

b) transfer management means for issuing to the transfer-executing subsystem a transfer request and for receiving a completion signal from the transfer-executing subsystem indicating completed transfer of the data set;

D) the emulated channel being further provided, upon receipt by the transfer management means of the completion signal,

i) for signaling to the transfer-initiating subsystem completed transfer of the first packet; and

ii) for receiving subsequent packets from the transfer-initiating subsystem and, for each, for signaling to the transfer-initiating subsystem completed transfer of each subsequent packet.

29. (original) A system as in claim 28, in which the direction of transfer of the data set is from the transfer-initiating subsystem to the target; and the transfer-initiating subsystem stores the data set in a memory space accessible to the intermediate subsystem, the system further comprising:

consolidation means, within the intermediate subsystem, for creating a copy of the data set, independent of action by the transfer-initiating subsystem, by accessing the memory space in which the transfer-initiating subsystem has stored the data set; and for identifying for the transfer-executing subsystem the copy of the data set as the data set to be transferred.

30. (original) A system as in claim 29, further comprising verification means for determining whether the transfer-initiating subsystem has changed the data set from the time when a first packet but before completion of the transfer by the transfer-executing subsystem and, if so, for creating and submitting to the transfer-executing subsystem for transfer to the target a copy of the changed data set.

31. (original) A system as in claim 28, in which the direction of transfer of the data set is from the target to the transfer-initiating subsystem, the system further comprising:

a buffer within the intermediate subsystem for storage of the data set transferred from the target; and

consolidation means within the intermediate subsystem for copying a corresponding portion of the data set to a respective one of the packets issued by the transfer-initiating subsystem upon completed transfer of the data set from the target.

32. (original) A system as in claim 28, in which the transfer-initiating subsystem is a software-implemented computer.

33. (original) A system as in claim 32, in which the transfer-initiating subsystem is a virtual machine.

34. (original) A system as in claim 28, in which:

the channel is a Universal Serial Bus (USB);

the target is a USB device and respective pipe; and

both the transfer-initiating subsystem and the transfer-executing subsystem generate and transfer packets according to the USB protocol.